

ATTORNEY DOCKET NO. W. DIEPSTRATEN 19-5-5

PATENT



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE  
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Application of: Wilhelmus J. M. Diepstraten, *et al.*

Serial No.: 09/213,984

Filed: December 17, 1998

Title: CONTEXT CONTROLLER HAVING CONTEXT-SPECIFIC EVENT  
SELECTION MECHANISM AND PROCESSOR EMPLOYING THE  
SAME

Group: 2154

Examiner: Larry D. Donaghue

RECEIVED

APR 08 2004

Technology Center 2100

Commissioner for Patents  
P. O. Box 1450  
Alexandria, VA 22313-1450

Mail Stop Appeal Brief - Patents

I hereby certify that this correspondence is being deposited with the United States Postal  
Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria,  
VA 22202, on 04/02/2004 (Date)

Stephane Ri

(Printed or typed name of person signing the certificate)

Stephane Ri

(Signature of the person signing the certificate)

Sir:

ATTENTION: Board of Patent Appeals and Interferences

**APPELLANT'S BRIEF UNDER 37 C.F.R. §1.192**

This is an appeal from a Final Rejection mailed on October 31, 2003, of Claims 1-22. The  
Appellants submit this Brief in triplicate as required by 37 C.F.R. §1.192(a), with the statutory fee  
of \$330.00 as set forth in 37 C.F.R. §1.17(c), and hereby authorize the Commissioner to charge any

04/07/2004 AWONDAF1 00000065 09213984

01 FC:1402

330.00 0P

additional fees connected with this communication or credit any overpayment to Deposit Account No. 08-2395.

This Brief contains these items under the following headings, and in the order set forth below in accordance with 37 C.F.R. §1.192(c):

- I. REAL PARTY IN INTEREST**
- II. RELATED APPEALS AND INTERFERENCES**
- III. STATUS OF THE CLAIMS**
- IV. STATUS OF THE AMENDMENTS**
- V. SUMMARY OF THE INVENTION**
- VI. ISSUES**
- VII. GROUPING OF THE CLAIMS**
- VIII. APPELLANTS' ARGUMENTS**
- IX. APPENDIX A - CLAIMS**

### **I. REAL PARTY IN INTEREST**

The real party in interest in this appeal are the joint Assignees Agere Systems, Incorporated and Choice-Intersil Microsystems, Incorporated.

### **II. RELATED APPEALS AND INTERFERENCES**

Related applications, application number 09/213,970 and application number 09/213,983, are presently under appeal and may have a bearing on the Board's decision in this appeal. No other appeals or interferences will directly affect or be directly affected by the Board's decision in this appeal.

### **III. STATUS OF THE CLAIMS**

Claims 1-22 are pending in this Application.

### **IV. STATUS OF THE AMENDMENTS**

The present Application was filed on December 17, 1998, containing Claims 1-22. The Appellants filed a first Amendment on November 11, 2002, in response to a first Examiner's Action mailed October 4, 2002. The first Amendment amended Claims 1, 8 and 15. No claims were canceled or added. In a first Final Rejection mailed on January 3, 2003, the Examiner indicated that the first Amendment did not place the Application in condition for allowance and maintained the previous rejection. In response, the Appellants filed a second Amendment on March 21, 2003, that amended Claims 1, 8 and 15 to place them in a better condition for appeal. The Examiner indicated

in a first Advisory Action mailed on April 16, 2003, that the second Amendment would require further search and consideration and would not be entered upon appeal.

In response, the Appellants filed a Request for Continued Examination on April 22, 2003, considering the second Amendment. The Examiner mailed a second Examiner's Action on May 9, 2003, addressing the Request for Continued Examination and maintaining the rejection of all the pending claims. The Appellants responded by filing a third Amendment on August 8, 2003, that amended Claims 1, 5, 8, 12, 15 and 19. The Examiner filed a second Final Rejection in response on October 31, 2003. The Appellants then filed a Request for Reconsideration on December 23, 2003, arguing for the pending Claims. On January 21, 2004, the Examiner mailed the second Advisory Action maintaining the rejection of Claims 1-22. The Appellants responded by mailing a Notice of Appeal on February 2, 2004.

## **V. SUMMARY OF THE INVENTION**

The present invention provides a context controller for managing multitasking in a processor and a method of operating the same. The present invention therefore introduces the broad concept of acknowledging only the events that are relevant to the context that is currently active (running). "Context," for purposes of the present invention, is defined as all processor state information (or any subset thereof, and such as register values) that would be of use in restoring the processor to a given state. An "event" is defined as a stimulus capable of causing the context controller to respond by switching from one foreground task to another. An exogenous event has its genesis outside the processor and can occur at any time. An endogenous event has its genesis inside the processor and is synchronized with the processor's clock.

The present invention provides an embodiment where all events are recorded but only those relevant to the currently-active context are acknowledged. Preferably, the currently-active context can control which events are acknowledged and which are (at least temporarily) ignored. In one embodiment, the context controller includes an event recorder that records occurrences of predetermined events and an event acknowledger, associated with the event recorder, that acknowledges ones of the events based on an identity of a currently-active context. In FIGURE 10 of the present application (set forth herein as ILLUSTRATION 1), illustrated is a schematic diagram of one embodiment of a circuit suitable for implementing event recording, event masking and event acknowledgment for each activation event, as well as management of a context activity bit, including initialization request and wait request logic where the details of event recording, masking and acknowledgment within the context controller may better be understood.

A generalized schematic fragment of a “slice” of a context controller event logic is presented for a single event including an ACT bit and WAIT function logic of the context associated with that event. In this diagram, all logic signals are considered to be asserted in the “high” true (logical one) state. An exogenous event signal 550 may be asserted with either polarity, so a programmable inversion function 560, under control of a software signal 551 may be provided to establish a high-true signal for internal use. Because this exogenous signal has an undetermined phase relationship with the internal clocks, a synchronizer 562 that synchronizes the input signal with a master clock rising edge Mr 517 prior to its internal use is employed. A plurality of sources may be used to set an event flip-flop 570 including a leading edge of a synchronized external signal 564, a leading edge of an internal source 566, or a software SIGNAL function 552 which designates this context and

event. These event sources are combined by an OR gate 568 whose output enables an event flip-flop 570 to be set at the master clock rising edge Mr 517.

FIG. 10

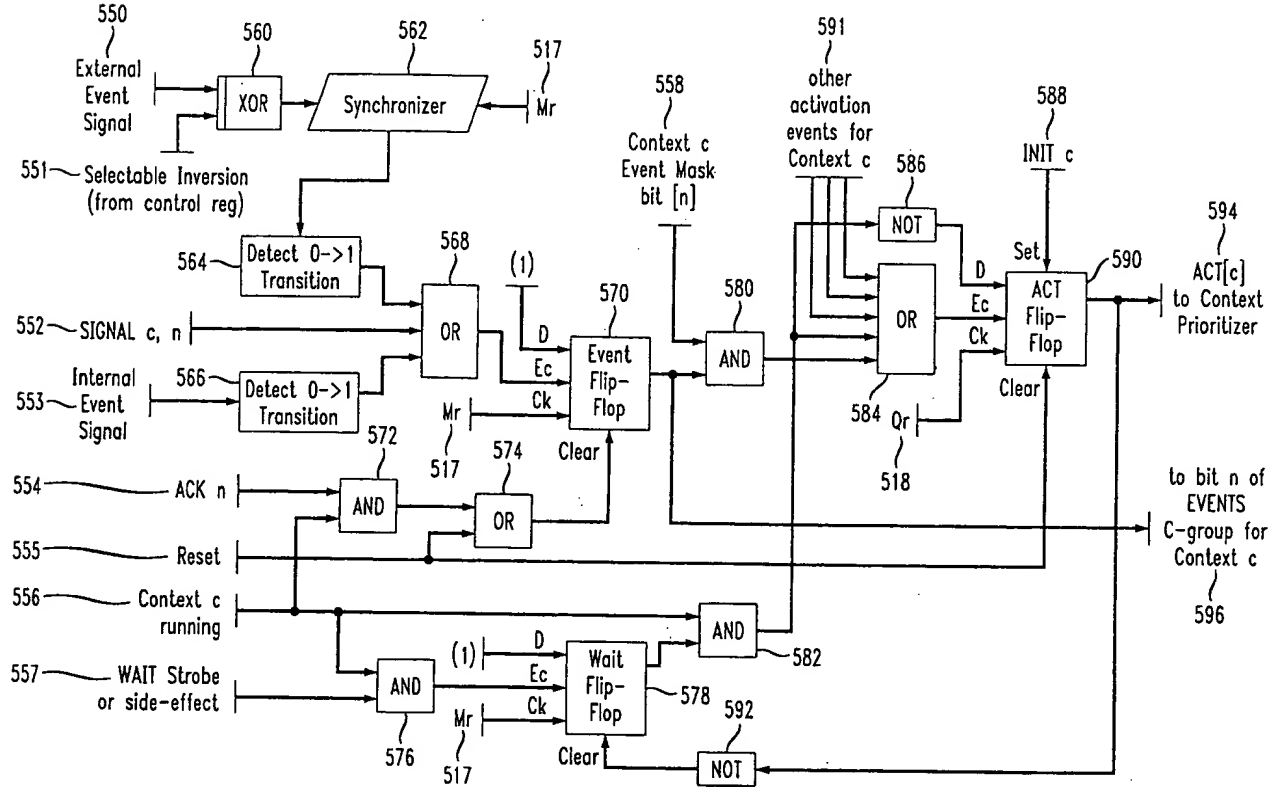


ILLUSTRATION 1

Because the event flip-flop D-input 570 is hardwired true (to a logical one as shown), negation of an event signal after setting the event flip-flop 570 does not rescind the event. The event flip-flop output 570 may be read by software as a bit in the event status register 94 and as a testable condition in an events condition signal group 596 if the processor provides instructions such as the SKPn of the illustrated embodiment (as described below). The event flip-flop 570 can be cleared either by a hardware reset 555 or an AND gate 572 output, whose ANDed inputs incorporate the execution of an ACK (acknowledge) function 554 for this event number while this context is running (a signal 556), applied through an OR gate 574.

An appropriate bit for this context event from the context's event mask register 94, event mask bit 558 is ANDed in an AND gate 580 with an event flip-flop output 570 and applied to the input of an ACT flip-flop 590 through an OR gate 584. The output of this AND gate 580 is also used when performing priority encoding of the context events for the VECTOR function, as is described in greater detail below. A masked event signal from the AND gate 580 is ORed in the OR gate 584 with the masked event signals from all other events associated with this context including a signal from the output gate of the wait logic through an AND gate 582.

A logical true output condition of the OR gate 584 enables the ACT flip-flop 590, allowing the ACT flip-flop 590 to be set to the output value of a NOT inverter 586 at the quadrature clock rising edge Qr 518. By using the output of the AND gate 582 and an inversion of the same signal through the NOT inverter 586, the Act flip-flop 590 D-input may be enabled. The ACT flip-flop 590 is set at the quadrature clock rising edge Qr 518 if one or more activation events are asserted, and no WAIT function was executed during the preceding instruction cycle. The ACT flip-flop 590 may also be set directly by execution of an INIT function 588 to this context, and cleared directly

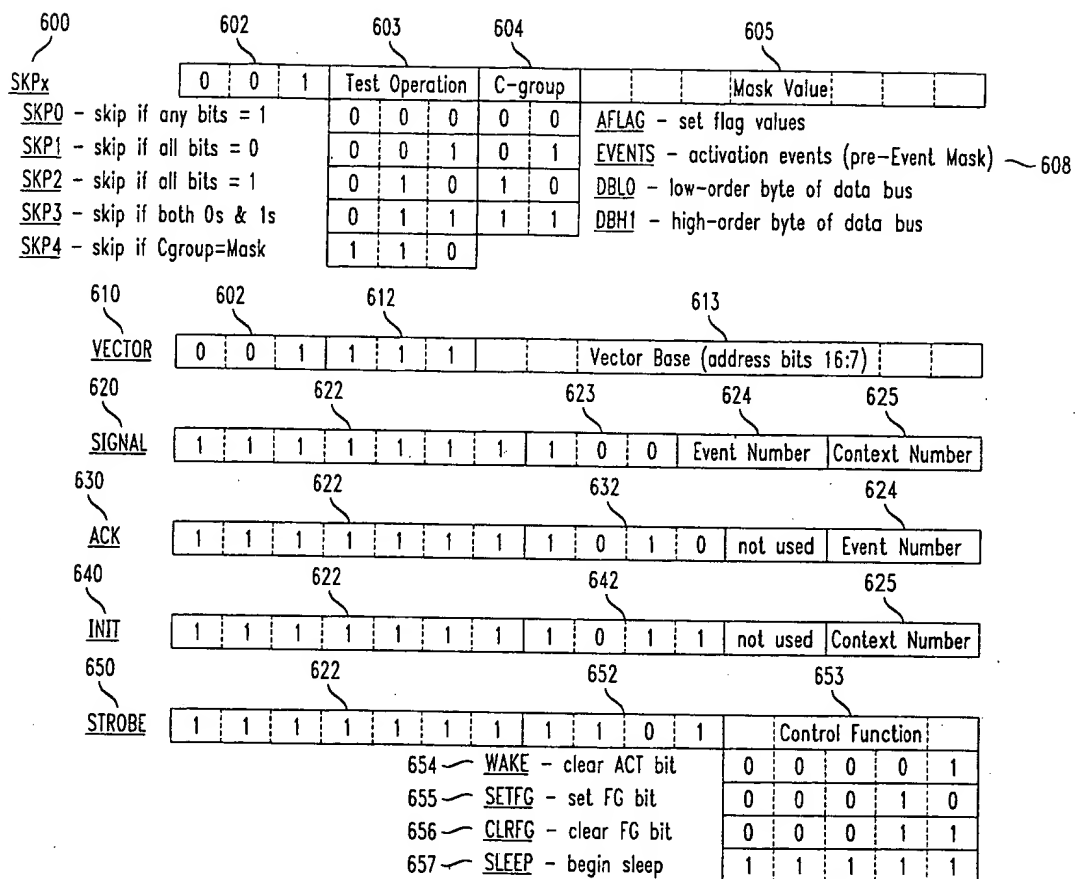
by a hardware reset signal 555. The ACT flip-flop output 590 is also used by the context priority logic and is inverted by a NOT inverter 592 to clear a WAIT flip-flop 578. The ACT flip-flop 590 is cleared through the NOT inverter 586 if a WAIT function was executed during the preceding instruction cycle whether or not any activation events are asserted. The WAIT flip-flop 578 is needed because a context may be preempted between executing a WAIT function and executing an instruction which follows the WAIT function. More discussion of the WAIT function and other components of the circuit of ILLUSTRATION 1 can be found at Page 60 of the present application.

Turning now to portions of FIGURE 11 of the present application (set forth herein as ILLUSTRATION 2), illustrated are field and bit assignments of machine instructions pertaining to context control and inter-context communication in the instruction set according to one embodiment of the present invention. A SIGNAL instruction 620 is used to implement an inter-context software signaling function. The SIGNAL instruction 620 is one of the processor control instructions based on the value of an extended opcode field 622 with a distinctive subdecode value 623. Two parameter fields are decoded within the context controller when a SIGNAL instruction is executed. A specified event number 624 identifies a particular event to assert among the events associated with a specified context number 625. All events may be the target of the SIGNAL instruction 620, but implementation details in particular instances of this context controller and connected event sources may make it difficult to allow the SIGNAL instruction 620 to assert certain conditions.

Regarding acknowledging, an ACK instruction 630 is formatted and decoded in a similar way to the SIGNAL instruction 620 but has only one parameter field. The ACK instruction 630 carries only an event number 624, because acknowledgment of a context's events is only permitted by code executing in the same context, so a context number parameter would be superfluous.



FIG. 11



## ILLUSTRATION 2

**2. Second Issue Presented for Review:**

Whether Claims 5 and 12 are properly rejected under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit in view of U.S. Patent No. 6,009,454 to Dummermuth, *et al.* (“Dummermuth”).

**3. Third Issue Presented for Review:**

Whether dependent Claims 6 and 13 are properly rejected under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit in view of U.S. Patent No. 5,239,652 to Seibert, *et al.* (“Seibert”).

**4. Fourth Issue Presented for Review:**

Whether Claims 7 and 14 are properly rejected under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit in view of U.S. Patent No. 6,256,659 to McLain, Jr. *et al.* (“McLain”).

**5. Fifth Issue Presented for Review:**

Whether Claims 15-18 and 22 are properly rejected under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit in view of U.S. Patent No. 5,713,038 to Motomura.

**6. Sixth Issue Presented for Review:**

Whether Claim 19 is properly rejected under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit in view of Motomura as applied to Claim 15 and in further view of Dummermuth.

**7. Seventh Issue Presented for Review:**

Whether Claim 20 is properly rejected under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit in view of Motomura as applied to Claim 15, and in further view of Seibert.

**8. Eight Issue Presented for Review:**

Whether Claim 21 is properly rejected under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit in view of Motomura as applied to Claim 15 and in further view of McLain.

**VII. GROUPING OF THE CLAIMS**

Claims 1-22 do not stand or fall together. Independent Claims 1, 8 and 15 form a first group and the dependent Claims form the following groups: Claims 2, 9 and 16 form a second group, Claims 3, 10 and 17 form a third group, Claims 4, 11 and 18 form a fourth group, and Claims 5, 12 and 19 form a fifth group, Claims 6, 13 and 20 form a sixth group, Claims 7, 14 and 21 form a seventh group, and Claim 22 forms an eighth group. Under the APPELLANT'S ARGUMENT section, the Appellants will explain why the Claims of the eight groups are believed to be separately patentable.

## **VIII. APPELLANTS' ARGUMENTS**

The inventions set forth in independent Claims 1, 8 and 15 and their respective dependent claims are not anticipated by nor are obvious over the references on which the Examiner relies.

### **A. Rejection of Claims 1 and 8 under 35 U.S.C. §102 and Claim 15 under U.S.C. §103**

The Examiner rejected Claims 1 and 8 under 35 U.S.C. §102(b) as being anticipated by Vaitzblit. Additionally, the Examiner has rejected Claim 15 as being unpatentable over Vaitzblit as applied to Claims 1 and 8 and in further view of Motomura. The Appellants respectfully disagree.

Vaitzblit is directed to a scheduler in a continuous media file server that supports multiple classes of tasks (isochronous, real-time and general-purpose) having diverse performance requirements. (Abstract). Vaitzblit discloses a communications system that includes the scheduler for these tasks. (Column 2, lines 54-62). The scheduler is hierarchical based in that a class of activities are scheduled first and then individual tasks within that class are scheduled. The class of isochronous tasks run periodically and are invoked by timer interrupts set for corresponding time periods. At the expiration of periodic timers, isochronous tasks arrive at the server and are inserted in an appropriate place on an isochronous ready queue. A unique periodic timer exist for each distinct period of admitted isochronous tasks. (Column 4, lines 31-47).

Whenever an isochronous task arrives at the server, the scheduler determines whether a currently running task needs to be preempted. For example, if the currently running task is a general-purpose task, it is preempted by the newly arrived isochronous task. If the currently running task is a real-time task, it will be preempted by the newly arrived isochronous task in a subsequent preemption window. If the currently running task is of the isochronous class, the scheduler

compares its priority to that of the newly arrived isochronous task. If the priority of the current task is lower, preemption occurs at the next preemption window by the isochronous task from the head of the queue. (Column 4, lines 48-59).

Vaitzblit does not specifically address acknowledging events and especially does not teach acknowledging events based on code of a currently-active context as recited in Claims 1 and 8. As discussed above, Vaitzblit teaches preempting tasks based on priority. The Examiner, however, asserts Vaitzblit teaches managing multitasking in a processor including acknowledging events based on code of a currently-active context as recited in Claims 1 and 8 since Vaitzblit teaches “code indicating a period.” (second Final Rejection, page 2). Though the Appellants do not find “code indicating a period” in the cited text of Vaitzblit, one can argue that the periodic information from the timers is used for preempting events. The periodic information from timers, however, is not code of currently-active context as recited in independent Claims 1 and 8. As stated previously, Vaitzblit teaches general purpose, real-time and the isochronous tasks. (Column 3, lines 27-33). Periodic information from a timer is not code of one of these tasks, and, more specifically, is not code of a currently active one of these tasks. Thus, Vaitzblit does not teach each and every element of independent Claims 1 and 8.

The Examiner has asserted in the second Advisory Action that the phrase “based on code” is an extremely broad phrase and that the tasks performed in the references would all have different code depending on the different classification. (second Advisory Action, page 2). Once again, Vaitzblit does not specifically address acknowledging events. Additionally, the Appellants point out that the limitation in Claims 1 and 8 is “based on code of a currently active context” not just “based on code.” Vaitzblit does not teach this limitation but actually teaches against it since Vaitzblit

teaches preempting a currently running task with another task having a higher priority. Thus, a currently running task would not be preempted by another task that is the same. On the contrary, preemption would only occur for a task having a higher priority. Vaitzblit therefore appears to teach that acknowledgment would only occur for those tasks having a higher priority instead of for those tasks that are relevant to a running context or, more specifically, based on code of a currently-active context. Since Vaitzblit does not teach each and every element of independent Claims 1 and 8, Vaitzblit does not anticipate Claims 1 and 8 and Claims dependent thereon. Claim 15 includes the subject matter of Claims 1 and 8 along with additional limitations. As discussed above, Vaitzblit does not teach each and every element of Claims 1 and 8. Additionally, Vaitzblit does not suggest each and every element of Claims 1 and 8 since Vaitzblit teaches against acknowledging events based on code of a currently-active context. Instead, Vaitzblit teaches preempting tasks based on priority. Vaitzblit, therefore, does not teach or suggest each and every element of Claims 1 and 8. Accordingly, Vaitzblit does not teach or suggest those limitations of Claim 15 for which it has been cited.

To teach the additional limitations of Claim 15, the Examiner cited Motomura. Motomura is directed to a microprocessor having a new register file for high speed, flexible, context switching. (Column 3, lines 15-17). The microprocessor is coupled to a memory and includes an instruction pipeline and a register file. At the time of context switching, the overhead required for the save/restore of the content stored in the register file can be reduced. (Abstract). Motomura, however, does not cure the deficiencies of Vaitzblit but has only been cited to teach a plurality of register sets and the interconnection of the plurality of register sets with an execution core. (second Final Rejection, page 5). The cited combination of Vaitzblit and Motomura, therefore, fails to teach

or suggest all of the elements of independent Claim 15 and does not provide a *prima facie* case of obviousness thereof. Accordingly, the Appellants respectfully traverse the Examiner's rejection of Claims 1 and 8 under 35 U.S.C. §102(b) and Claim 15 under 35 U.S.C. §103(a) and respectfully request that the Board of Patent Appeals and Interferences reverse the Examiner's Final Rejection of Claims 1, 8 and 15.

**B. Rejection of Claims 2 and 9 under 35 U.S.C. §102 and Claim 16 under U.S.C. §103**

The Examiner rejected Claims 2 and 9 under 35 U.S.C. §102(b) as being anticipated by Vaitzblit and Claim 16 as being unpatentable over Vaitzblit as applied to Claims 1 and 8 and in further view of Motomura. The above argument establishing that Vaitzblit does not anticipate independent Claims 1 and 8 and Vaitzblit combined with Motomura does not render obvious the invention of independent Claim 15 is incorporated herein by reference. Dependent Claims 2, 9 and 16 additionally require masking others of the events as a function of each context, and thereby introduce patentably distinct elements in addition to the elements recited in Claims 1, 8 and 15, respectively. Vaitzblit nor the cited combination of Vaitzblit and Motomura, however, teach or suggest masking others of the events as a function of each context in combination with the limitations of Claims 1, 8 and 15, respectively. Thus, Vaitzblit does not teach each and every element of Claims 2 and 9 and the cited combination of Vaitzblit and Motomura does not establish a *prima facie* case of obviousness of dependent Claim 16. Claims 2 and 9, therefore, are not anticipated by Vaitzblit and Claim 16 is not obvious over Vaitzblit and Motomura. Accordingly, the Appellants respectfully request that the Board of Patent Appeals and Interferences reverse the Examiner's Final Rejection thereof.

**C. Rejection of Claims 3 and 10 under 35 U.S.C. §102 and Claim 17 under U.S.C. §103**

The Examiner rejected Claims 3 and 10 under 35 U.S.C. §102(b) as being anticipated by Vaitzblit and Claim 17 as being unpatentable over Vaitzblit as applied to Claims 1 and 8 and in further view of Motomura. The above argument establishing that Vaitzblit does not anticipate independent Claims 1 and 8 and Vaitzblit combined with Motomura does not render obvious the invention of independent Claim 15 is incorporated herein by reference. Dependent Claims 3, 10 and 17 additionally require the event recorder is embodied in at least one flip-flop within the context controller, and thereby introduce patentably distinct elements in addition to the elements recited in Claims 1, 8 and 15, respectively. Vaitzblit nor the cited combination of Vaitzblit and Motomura, however, teach or suggest an event recorder embodied in at least one flip-flop within the context controller in combination with the limitations of Claims 1, 8 and 15, respectively. Thus, Vaitzblit does not teach each and every element of Claims 3 and 10 and the cited combination of Vaitzblit and Motomura does not establish a *prima facie* case of obviousness of dependent Claim 17. Claims 3 and 10, therefore, are not anticipated by Vaitzblit and Claim 17 is not obvious over Vaitzblit and Motomura. Accordingly, the Appellants respectfully request that the Board of Patent Appeals and Interferences reverse the Examiner's Final Rejection thereof.

**D. Rejection of Claims 4 and 11 under 35 U.S.C. §102 and Claim 18 under U.S.C. §103**

The Examiner rejected Claims 4 and 11 under 35 U.S.C. §102(b) as being anticipated by Vaitzblit and Claim 18 as being unpatentable over Vaitzblit as applied to Claims 1 and 8 and in further view of Motomura. The above argument establishing that Vaitzblit does not anticipate



independent Claims 1 and 8 and Vaitzblit combined with Motomura does not render obvious the invention of independent Claim 15 is incorporated herein by reference. Dependent Claims 4, 11 and 18 additionally require activating contexts corresponding to foreground tasks based on priority and in response to the events and cyclicly activating contexts corresponding to the background tasks subject to activation of the contexts corresponding to the foreground tasks, and thereby introduce patentably distinct elements in addition to the elements recited in Claims 1, 8 and 15, respectively. Vaitzblit nor the cited combination of Vaitzblit and Motomura, however, teach or suggest activating contexts corresponding to foreground tasks based on priority and in response to the events and cyclicly activating contexts corresponding to the background tasks subject to activation of the contexts corresponding to the foreground tasks in combination with the limitations of Claims 1, 8 and 15, respectively. Thus, Vaitzblit does not teach each and every element of Claims 4 and 11 and the cited combination of Vaitzblit and Motomura does not establish a *prima facie* case of obviousness of dependent Claim 18. Claims 4 and 11, therefore, are not anticipated by Vaitzblit and Claim 18 is not obvious over Vaitzblit and Motomura. Accordingly, the Appellants respectfully request that the Board of Patent Appeals and Interferences reverse the Examiner's Final Rejection thereof.

**E. Rejection of Claim 5, 12 and 19 under 35 U.S.C. §103**

The Examiner has rejected Claims 5 and 12 under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit and in further view of Dummermuth. Additionally, the Examiner has rejected Claim 19 as being unpatentable over Vaitzblit and Motomura as applied to Claim 15 and in further view of Dummermuth. The above argument establishing that Vaitzblit does not anticipate independent

Claims 1 and 8 and Vaitzblit combined with Motomura does not render obvious the invention of independent Claim 15 is incorporated herein by reference. Dependent Claims 5, 12 and 19 additionally require activating contexts corresponding to background tasks based on numbers of instructions executed by each of the background tasks, wherein each of the background tasks accomplishes an equal amount of work before a cycle of background processing repeats, and thereby introduce patentably distinct elements in addition to the elements recited in Claims 1, 8 and 15, respectively. The Examiner has cited Dummermuth to address the patentably distinct elements included in Claims 5, 12 and 19. (second Final Rejection, page 3).

Dummermuth is directed to a multi-tasking operating system which is provided by recognizing that both ladder-type and state-type programs can be considered as simply a collection of individual instructions linked together by an implicit pointer list. (Column 2, lines 48-53). Regarding Claims 1, 8 and 15, Dummermuth does not cure the deficiencies of Vaitzblit or the combination of Vaitzblit and Motomura since Dummermuth does not teach or suggest managing multitasking in a processor including acknowledging events based on code of a currently-active context. Instead, Dummermuth teaches executing programs according to a fixed number of allocated instructions. (Abstract).

Additionally, as stated by the Examiner, the cited combination of Vaitzblit, Motomura and Dummermuth does not show each of the background tasks accomplishes an equal amount of work before a cycle of background processing repeats. The Examiner asserts that it would be obvious to modify Vaitzblit to make each background task accomplish an equal amount of work before a cycle of background processing repeats since Vaitzblit suggest processing under a guaranteed time frame. (second Final Rejection , page 3). The Appellants, however, do not understand how one skilled in

the art would be so motivated since Vaitzblit teaches time slice scheduling. (Column 5, lines 1-33). On the contrary, a guaranteed time frame would appear to teach away from background tasks accomplishing an equal amount of work before a cycle of background processing repeats since a guaranteed time frame may not allow an equal amount of work before repeating. This is especially true considering the priority preempting of Vaitzblit.

Thus, the cited combination of Vaitzblit and Dummermuth does not teach each and every element of Claims 5 and 12 and the cited combination of Vaitzblit, Motomura and Dummermuth does not establish a *prima facie* case of obviousness of dependent Claim 19. Claims 5 and 12, therefore, are not obvious over the cited combination of Vaitzblit and Dummermuth and Claim 18 is not obvious over the cited combination of Vaitzblit, Motomura and Dummermuth. Accordingly, the Appellants respectfully request that the Board of Patent Appeals and Interferences reverse the Examiner's Final Rejection thereof.

**F. Rejection of Claims 6, 13 and 20 under 35 U.S.C. §103**

The Examiner has rejected Claims 6 and 13 under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit in view of Seibert. Additionally, the Examiner has rejected Claim 20 as being unpatentable over Vaitzblit and Motomura as applied to Claim 15 and in further view of Seibert. The above argument establishing that Vaitzblit does not anticipate independent Claims 1 and 8 and Vaitzblit combined with Motomura does not render obvious the invention of independent Claim 15 is incorporated herein by reference. Seibert is directed to reducing the power consumption of a computer by determining when the central processing unit (CPU) is not actively processing and generating a power-off signal to a control logic circuit. (Abstract). The control logic circuit controls

the CPU, which is powered by a power supply, such that the CPU is completely disconnected from the power supply and brought into substantially full power periodically at predetermined intervals by interrupts when the CPU is in an inactive state. (Column 3, lines 27-35).

Seibert has not been cited to cure the deficiencies of Vaitzblit and Motomura but has only been offered to teach the patentably distinct elements of Claims 6, 13, and 20 that are in addition to the elements recited in Claims 1, 8 and 15, respectively. More specifically, Seibert has been cited to show placing a processor in an idle state when all foreground and background task are inactive. (second Final Rejection, page 4). Vaitzblit and Seibert nor Vaitzblit, Motomura and Seibert teach or suggest placing a processor in an idle state when all foreground and background task are inactive in combination with the limitations of Claims 1 and 8 and Claim 15, respectively. Thus, the cited combinations of Vaitzblit, Motomura and Seibert do not establish a *prima facie* case of obviousness of dependent Claims 6, 13 and 20. Claims 6 and 13, therefore, are nonobvious over the cited combination of Vaitzblit and Seibert and Claim 20 is nonobvious over the cited combination of Vaitzblit, Motomura and Seibert. Accordingly, the Appellants respectfully request that the Board of Patent Appeals and Interferences reverse the Examiner's Final Rejection thereof.

**G. Rejection of Claims 7, 14 and 21 under 35 U.S.C. §103**

The Examiner has rejected Claims 7 and 14 under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit and in further view of McLain. Additionally, the Examiner has rejected Claim 21 as being unpatentable over Vaitzblit and Motomura as applied to Claim 15 and in further view of McLain. The above argument establishing that Vaitzblit does not anticipate independent Claims 1 and 8 and Vaitzblit combined with Motomura does not render obvious the invention of independent

Claim 15 is incorporated herein by reference. Dependent Claims 7, 14 and 21 additionally require that the foreground task controller is adapted to activate a context corresponding to a particular foreground task by vectoring to a software-selectable memory location, and thereby introduce patentably distinct elements in addition to the elements recited in Claims 1, 8 and 15, respectively. The cited combination of Vaitzblit and McLain nor the cited combination of Vaitzblit, Motomura and McLain teach or suggest a foreground task controller that is adapted to activate a context corresponding to a particular foreground task by vectoring to a software-selectable memory location in combination with the limitations of Claims 1, 8 and 15. Thus, Vaitzblit and McLain nor Vaitzblit, Motomura and McLain establish a *prima facie* case of obviousness of dependent Claims 7, 14 and 21, respectively. Accordingly, Claims 7, 14 and 21 are nonobvious over the cited combinations of Vaitzblit, Motomura and McLain and the Appellants respectfully request that the Board of Patent Appeals and Interferences reverse the Examiner's Final Rejection thereof.

#### **H. Rejection of Claim 22 under 35 U.S.C. §103**

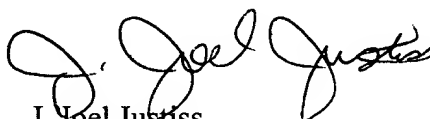
The Examiner has rejected Claim 22 under 35 U.S.C. §103(a) as being unpatentable over Vaitzblit and in further view of Motomura. The above argument establishing that Vaitzblit, Motomura and McLain do not render obvious the invention of independent Claim 15 is incorporated herein by reference. Dependent Claim 22 additionally requires the processor forms a portion of a general-purpose computer, and thereby introduces patentably distinct elements in addition to the elements recited in independent Claim 15. Vaitzblit and Motomura, however, do not teach or suggest the processor forms a portion of a general-purpose computer in combination with the limitations of Claim 15. Thus, Vaitzblit, Motomura and McLain do not establish a *prima facie* case

of obviousness of dependent Claim 22. Accordingly, Claim 22 is nonobvious over Vaitzblit, Motomura and McLain and the Appellants respectfully request that the Board of Patent Appeals and Interferences reverse the Examiner's Final Rejection thereof.

In view of the foregoing remarks, the cited references do not support the Examiner's rejection of the pending Claims under 35 U.S.C. §102(b) and 35 U.S.C. §103(a). Accordingly, the Appellants respectfully request that the Board of Patent Appeals and Interferences reverse the Examiner's Final Rejection of Claims 1-22.

Respectfully submitted,

**HITT GAINES, P.C.**

  
J. Joel Justiss  
Registration No. 48,981

Dated: 4/2/04

## **IX. APPENDIX A - CLAIMS**

1. A context controller for managing multitasking in a processor, comprising:  
an event recorder that records occurrences of predetermined events; and  
an event acknowledger, associated with said event recorder, that acknowledges ones of said events based on code of a currently-active context.
2. The context controller as recited in Claim 1 further comprising an event masker, associated with said event recorder, said event acknowledger and each context executing on said processor, that masks others of said events as a function of said each context.
3. The context controller as recited in Claim 1 wherein said event recorder is embodied in at least one flip-flop within said context controller.
4. The context controller as recited in Claim 1 further comprising;  
a foreground task controller that activates contexts corresponding to foreground tasks based on priority and in response to said events; and  
a background task controller that cyclicly activates contexts corresponding to said background tasks subject to activation of said contexts corresponding to said foreground tasks.
5. The context controller as recited in Claim 1 further comprising a background task controller that activates contexts corresponding to background tasks based on numbers of instructions executed by each of said background tasks, wherein said each of said background tasks accomplishes an equal amount of work before a cycle of background processing repeats.
6. The context controller as recited in Claim 1 wherein said context controller places said processor in an idle state when all foreground and background tasks are inactive.

7. The context controller as recited in Claim 1 further comprising a foreground task controller adapted to activate a context corresponding to a particular foreground task by vectoring to a software-selectable memory location.

8. A method of managing multitasking in a processor, comprising the steps of:

recording occurrences of predetermined events; and

acknowledging ones of said events based on code of a currently-active context.

9. The method as recited in Claim 8 further comprising the step of masking others of said events as a function of each context executing on said processor.

10. The method as recited in Claim 8 wherein said step of recording comprises the step of changing a state of at least one flip-flop within said context controller.

11. The method as recited in Claim 8 further comprising the steps of:

activating contexts corresponding to foreground tasks based on priority and in response to said events; and

cyclicly activating contexts corresponding to said background tasks subject to activation of said contexts corresponding to said foreground tasks.

12. The method as recited in Claim 8 further comprising the step of activating contexts corresponding to background tasks based on numbers of instructions executed by each of said background tasks, wherein said each of said background tasks accomplishes an equal amount of work before a cycle of background processing repeats.

13. The method as recited in Claim 8 further comprising the step of placing said processor in an idle state when all foreground and background tasks are inactive.



14. The method as recited in Claim 8 further comprising the step of activating a context corresponding to a particular foreground task by vectoring to a software-selectable

15. A processor, comprising:

an instruction decoder that decodes instructions received into said processor and corresponding to a plurality of tasks;

a plurality of register sets, corresponding to said plurality of tasks, that contain operands to be manipulated;

an execution core, coupled to said instruction decoder and said plurality of register sets, that executes instructions corresponding to an active one of said plurality of tasks to manipulate ones of said operands; and

a context controller for managing multitasking in said processor, including:

an event recorder that records occurrences of predetermined events; and

an event acknowledger, associated with said event recorder, that acknowledges ones of said events based on code of a currently-active context.

16. The processor as recited in Claim 15 wherein said context controller further includes an event masker, associated with said event recorder, said event acknowledger and each context executing on said processor, that masks others of said events as a function of said each context.

17. The processor as recited in Claim 15 wherein said event recorder is embodied in at least one flip-flop within said context controller.

18. The processor as recited in Claim 15 wherein said context controller further includes:

a foreground task controller that activates contexts corresponding to foreground tasks based on priority and in response to said events; and

a background task controller that cyclicly activates contexts corresponding to said background tasks subject to activation of said contexts corresponding to said foreground tasks.

19. The processor as recited in Claim 15 wherein said context controller further includes a background task controller that activates contexts corresponding to background tasks based on numbers of instructions executed by each of said background tasks, wherein said each of said background tasks accomplishes an equal amount of work before a cycle of background processing repeats.

20. The processor as recited in Claim 15 wherein said context controller places said processor in an idle state when all foreground and background tasks are inactive.

21. The processor as recited in Claim 15 wherein said context controller further includes a foreground task controller adapted to activate a context corresponding to a particular foreground task by vectoring to a software-selectable memory location.

22. The processor as recited in Claim 15 wherein said processor forms a portion of a general-purpose computer.